

Reinforcement Learning

Exercise 2: Value Functions, Dynamic Programming and Optimal Policies

Alexander Mattick

Overview

Exercise Content

Week	Date	Topic	Material	Who?
1	22.04.		<i>no exercises</i>	
2	29.04.	MDPs (slides)	ex1.pdf	Nico
3	06.05.	– no exercises –		
4	13.05.	Dynamic Programming (slides)	ex2.pdf, ex2_skeleton.zip	Alex
5	20.05.	OpenAI Gym, PyTorch-Intro (slides) TD-Learning (slides)		Nico
6	27.05.	TD-Control (slides)		Nico
7	03.06.	Intermediate exam		
8	10.06.		<i>no exercises</i>	
9	17.06.	DQN (slides)		Nico
10	24.06.	VPG (slides)		Alex
11	01.07.	A2C (slides)		Nico
12	08.07.	Multi-armed Bandits (slides)		Alex
13	15.07.	RND/ICM (slides)		Alex
14	22.07.	MCTS (slides)		Alex



Dynamic Programming



Markov Decision Processes

Recap

- We need a controller that helps us select the actions to maximize expected cumulative reward
 - So-called: **Expected return** or **value**
- A policy π represents this controller:
 - π determines the agent's behavior, i.e., its way of acting
 - π is a mapping from state space \mathcal{S} to action space \mathcal{A} , i.e., $\pi : \mathcal{S} \mapsto \mathcal{A}$
 - Two types of policies:
 - Deterministic policy: $a = \pi(s)$.
 - Stochastic policy: $\pi(a | s) = \mathbb{P}[A_t = a | S_t = s]$
- **Goal:** find a policy that maximizes the expected return!
 - We denote the optimal policy π for a given MDP as π^*

Markov Decision Processes

The Value Function

(State-)Value function

$$V_{\pi}(s) = \mathbb{E}_{\pi}[G_t | s_t = s] = \mathbb{E}_{\pi} \left[\sum_{t=0}^{\infty} \gamma^t r_t | s_t = s \right]$$

- “Expected return following policy π from state s ”

Action-value function/Q-function

$$Q_{\pi}(s, a) = \mathbb{E}_{\pi}[G_t | s_t = s, a_t = a] = \mathbb{E}_{\pi} \left[\sum_{t=0}^{\infty} \gamma^t r_t | s_t = s, a_t = a \right]$$

- “Expected return of doing action a in state s and following policy π afterwards”

Dynamic Programming

Introduction

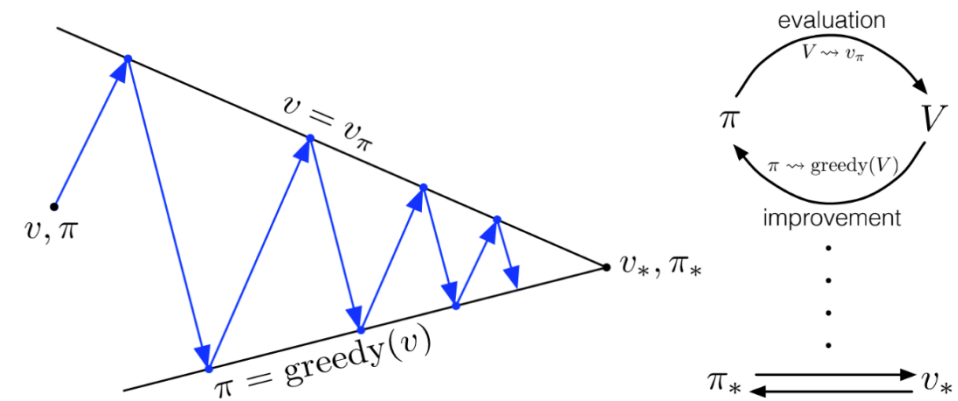
- Limited utility in practical reinforcement learning, but theoretical importance
 - Why?
- **Idea:** Use value functions to organize and structure the search for good policies
 - We can easily obtain optimal policies once we have found the optimal value function (and vice versa)
 - Founded on the Bellman optimality equation(s)

Bellman-Optimality Equation

$$V_{\pi^*}(s) = \max_a Q_{\pi^*}(s, a) = \max_a \mathbb{E}_{\pi^*}[r_t + \gamma V_{\pi^*}(s)]$$

Four key concepts

- Policy Evaluation
- Policy Improvement
- (Generalized) Policy Iteration
- Value Iteration



Sutton, R. S., & Barto, A. G. (2018). Reinforcement learning: An introduction. MIT press.

Policy Evaluation

- Given a policy π and the environment dynamics, we can easily compute the value of state:

$$V_{\pi}(s) = \mathbb{E}_{\pi}[G_t \mid s_t = s] = \dots = \sum_a \pi(a|s) \sum_{s', r} P(s', r \mid s, a) [r + \gamma V_{\pi}(s')]$$

- System of *#states* linear equations with *#states* unknowns
 - Can be solved straightforwardly
 - For our purposes, we solve it iteratively

Iterative Policy Evaluation, for estimating $V \approx v_{\pi}$

Input π , the policy to be evaluated

Algorithm parameter: a small threshold $\theta > 0$ determining accuracy of estimation

Initialize $V(s)$ arbitrarily, for $s \in \mathcal{S}$, and $V(\text{terminal})$ to 0

Loop:

$\Delta \leftarrow 0$

Loop for each $s \in \mathcal{S}$:

$v \leftarrow V(s)$

$V(s) \leftarrow \sum_a \pi(a|s) \sum_{s', r} p(s', r \mid s, a) [r + \gamma V(s')]$

$\Delta \leftarrow \max(\Delta, |v - V(s)|)$

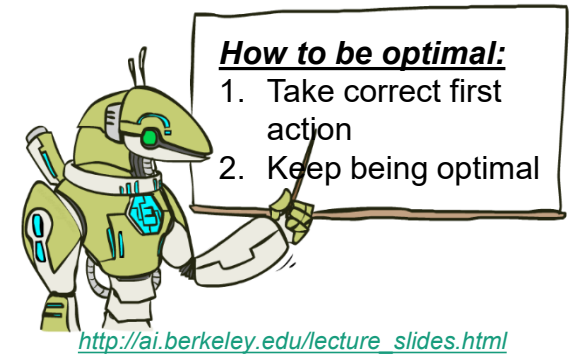
until $\Delta < \theta$

Policy Improvement

- Given a policy π and its value function (and the environment dynamics), greedily take the action that looks good in the short term

$$\pi'(s) = \arg \max_a Q_\pi(s, a)$$

- Suppose $\pi' = \pi$, then π' fulfils the Bellman optimality equation in all states
 - Therefore: We found the optimal policy



Policy Iteration

Policy Iteration (using iterative policy evaluation) for estimating $\pi \approx \pi_*$

1. Initialization

$V(s) \in \mathbb{R}$ and $\pi(s) \in \mathcal{A}(s)$ arbitrarily for all $s \in \mathcal{S}$; $V(\text{terminal}) \doteq 0$

2. Policy Evaluation

Loop:

$\Delta \leftarrow 0$

Loop for each $s \in \mathcal{S}$:

$v \leftarrow V(s)$

$V(s) \leftarrow \sum_{s',r} p(s',r|s,\pi(s)) [r + \gamma V(s')]$

$\Delta \leftarrow \max(\Delta, |v - V(s)|)$

until $\Delta < \theta$ (a small positive number determining the accuracy of estimation)

3. Policy Improvement

policy-stable \leftarrow *true*

For each $s \in \mathcal{S}$:

old-action $\leftarrow \pi(s)$

$\pi(s) \leftarrow \operatorname{argmax}_a \sum_{s',r} p(s',r|s,a) [r + \gamma V(s')]$

If *old-action* $\neq \pi(s)$, then *policy-stable* \leftarrow *false*

If *policy-stable*, then stop and return $V \approx v_*$ and $\pi \approx \pi_*$; else go to 2

Value Iteration

- **Drawback of Policy Iteration:** We must do a full Policy Evaluation procedure for every step, which is costly!
- We can also truncate this:
 - If we stop the policy evaluation after just one sweep, this is called **Value Iteration**
 - Surprisingly, this corresponds to translating the Bellman optimality equation into an update rule
 - We can also drop the policy improvement step because we are only interested in the final policy
 - Downside of this: More iteration to convergence needed

Value Iteration, for estimating $\pi \approx \pi_*$

Algorithm parameter: a small threshold $\theta > 0$ determining accuracy of estimation

Initialize $V(s)$, for all $s \in \mathcal{S}^+$, arbitrarily except that $V(\text{terminal}) = 0$

Loop:

```
|  $\Delta \leftarrow 0$   
| Loop for each  $s \in \mathcal{S}$ :  
|    $v \leftarrow V(s)$   
|    $V(s) \leftarrow \max_a \sum_{s',r} p(s', r | s, a) [r + \gamma V(s')]$   
|    $\Delta \leftarrow \max(\Delta, |v - V(s)|)$   
until  $\Delta < \theta$ 
```

Output a deterministic policy, $\pi \approx \pi_*$, such that

$$\pi(s) = \arg \max_a \sum_{s',r} p(s', r | s, a) [r + \gamma V(s')]$$

Dynamic Programming

Summary

- Policy Evaluation
 - Given policy π , compute its (approximate) value function for (part of) the state space
- Policy Improvement
 - Given value function $V_\pi(s)$, extract the greedy policy π' with $V_{\pi'}(s) \geq V_\pi(s)$
- (Generalized) Policy Iteration
 - Repeat until convergence (policy doesn't change after improvement, i.e., Bellman optimality equation holds)
 - Do x steps of Policy Evaluation
 - Do Policy Improvement
- Value Iteration
 - Special case of Policy Iteration with 1 Policy Evaluation step
 - Converged when change in value estimates smaller than some threshold
 - Policy Improvement step only as the last step

Thank you for your attention!